

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2003-256067
 (43)Date of publication of application : 10.09.2003

(51)Int.Cl.

G06F 1/04
G06F 9/46

(21)Application number : 2002-055234
 (22)Date of filing : 01.03.2002

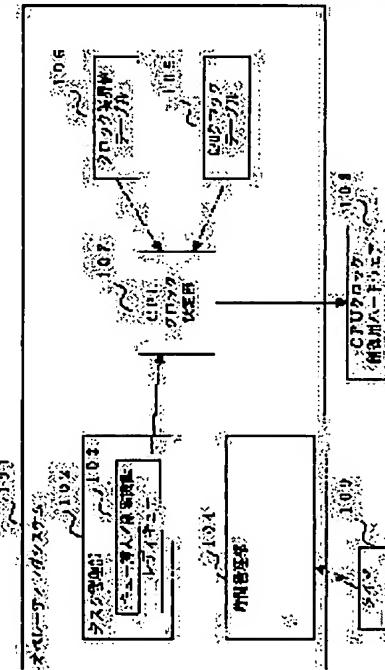
(71)Applicant : MITSUBISHI ELECTRIC CORP
 (72)Inventor : KUROSAWA HISAYOSHI

(54) POWER SAVING CONTROL SYSTEM, POWER SAVING CONTROL METHOD, PROGRAM AND RECORDING MEDIUM

(57)Abstract:

PROBLEM TO BE SOLVED: To provide a power saving control system for deciding operation clock of a processor by considering the number of tasks in waiting states in addition to priorities of tasks.

SOLUTION: The power saving control system has a priority table for keeping a plurality of priorities and the number of tasks which are waiting for execution according to the respective priority, a CPU clock table 105 for keeping a plurality of operation clock values capable of setting to the processor, a clock boundary value table 106 for keeping a reference value for deciding the operation clock of the processor and a clock decision part 107 for selecting the operation clock from the CPU clock table 105 by calculating loading value of the processor with the plurality of priorities and the number of tasks and by using the calculated loading value and the reference value and for setting the selected operation clock to the processor.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

(19)日本国特許庁 (JP)

(12) 公開特許公報 (A)

(11)特許出願公開番号

特開2003-256067

(P2003-256067A)

(43)公開日 平成15年9月10日 (2003.9.10)

(51)Int.Cl.⁷

G 0 6 F 1/04
9/46

識別記号

3 0 1
3 4 0

F I

G 0 6 F 1/04
9/46

マーク* (参考)

3 0 1 C 5 B 0 7 9
3 4 0 B 5 B 0 9 8
3 4 0 D

審査請求 未請求 請求項の数10 O.L. (全 14 頁)

(21)出願番号 特願2002-55234(P2002-55234)

(22)出願日 平成14年3月1日 (2002.3.1)

(71)出願人 000006013

三菱電機株式会社

東京都千代田区丸の内二丁目2番3号

(72)発明者 黒澤 寿好

東京都千代田区丸の内二丁目2番3号 三菱電機株式会社内

(74)代理人 100099461

弁理士 溝井 章司 (外5名)

F ターム (参考) 5B079 BA01 BB01 BC01

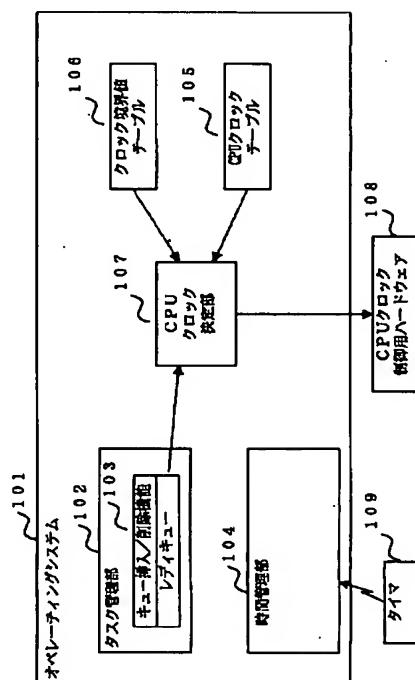
5B098 FF03 GA04 GC08 GD01 GD14

(54)【発明の名称】 省電力制御方式及び省電力制御方法及びプログラム及び記録媒体

(57)【要約】

【課題】 タスクの優先度に加え、待ち状態にあるタスクの数を考慮してプロセッサの動作クロックを決定する省電力制御方式を提供する。

【解決手段】 省電力制御方式は、複数の優先度と、上記複数の優先度それぞれによる実行を待つタスクの数とを保持する優先度テーブルと、プロセッサへ設定することができる複数の動作クロック値を保持するCPUクロックテーブル105と、プロセッサの動作クロックを決めるための基準値を保持するクロック境界値テーブル106と、上記優先度テーブルに保持する複数の優先度とタスクの数とを用いて、プロセッサにかかる負荷値を算出し、算出した負荷値と基準値とを用いて、CPUクロックテーブル105から動作クロック値を選択し、選択した動作クロック値を上記プロセッサへ設定するクロック決定部107とを備える。



【特許請求の範囲】

【請求項1】 複数のタスクを動作させるプロセッサの動作クロックを制御する省電力制御方式において、複数の優先度と、上記複数の優先度それぞれによる実行を待つタスクの数とを保持する優先度テーブルと、上記プロセッサへ設定することができる複数の動作クロック値を保持するクロックテーブルと、上記優先度テーブルから上記複数の優先度と上記タスクの数とを読み出し、読み出した複数の優先度とタスクの数とを用いて、上記プロセッサにかかる負荷値を算出し、算出した負荷値を用いて、上記クロックテーブルから動作クロック値を選択し、選択した動作クロック値を上記プロセッサへ設定するクロック決定部とを備えることを特徴とする省電力制御方式。

【請求項2】 上記省電力制御方式は、さらに、上記プロセッサの動作クロックを決めるための基準値を保持する基準値テーブルを備え、上記クロック決定部は、上記基準値テーブルから基準値を読み出し、上記基準値と上記負荷値とを用いて、動作クロックを選択することを特徴とする請求項1記載の省電力制御方式。

【請求項3】 上記優先度テーブルは、さらに、上記複数の優先度それぞれに対応する重み付け値を保持し、上記クロック決定部は、上記複数の優先度と上記タスクの数と上記重み付け値とを用いて負荷値を算出することを特徴とする請求項1または2記載の省電力制御方式。

【請求項4】 上記優先度テーブルは、さらに、上記複数の優先度それぞれに対応する設定クロック値を保持し、上記クロック決定部は、上記負荷値を算出する場合に、上記設定クロック値を用いることを特徴とする請求項1から3いずれかに記載の省電力制御方式。

【請求項5】 上記省電力制御方式は、さらに、実行を要求するタスク名と実行を終了したタスク名とのいずれかを入力し、入力したタスク名から優先度を取得し、取得した優先度を用いて、上記優先度テーブルに保持するタスクの数を更新するタスク管理部を備えることを特徴とする請求項1から4いずれかに記載の省電力制御方式。

【請求項6】 上記省電力制御方式は、上記タスク管理部が優先度テーブルに保持するタスク数を更新した場合に、上記クロック決定部を起動することを特徴とする請求項5記載の省電力制御方式。

【請求項7】 上記省電力制御方式は、所定の周期で上記クロック決定部を起動することを特徴とする請求項1から6いずれかに記載の省電力制御方式。

【請求項8】 複数のタスクを動作させるプロセッサの動作クロックを制御する省電力制御方法において、複数の優先度と、上記複数の優先度それぞれに対応する優先度による実行を待つタスクの数とを優先度テーブル

へ格納する優先度テーブル格納工程と、上記プロセッサへ設定することができる複数の動作クロック値をクロックテーブルへ格納するクロックテーブル格納工程と、上記優先度テーブルから上記複数の優先度と上記タスクの数とを読み出し、読み出した複数の優先度とタスクの数とを用いて、上記プロセッサにかかる負荷値を算出し、算出した負荷値を用いて、上記クロックテーブルから動作クロック値を選択し、選択した動作クロック値を上記プロセッサへ設定するクロック決定工程とを備えることを特徴とする省電力制御方法。

【請求項9】 複数のタスクを動作させるプロセッサの動作クロックを制御する省電力制御を計算機で実行させるプログラムにおいて、複数の優先度と、上記複数の優先度それぞれに対応する優先度による実行を待つタスクの数とを優先度テーブルへ格納する優先度テーブル格納処理と、上記プロセッサへ設定することができる複数の動作クロック値をクロックテーブルへ格納するクロックテーブル格納処理と、

上記優先度テーブルから上記複数の優先度と上記タスクの数とを読み出し、読み出した複数の優先度とタスクの数とを用いて、上記プロセッサにかかる負荷値を算出し、算出した負荷値を用いて、上記クロックテーブルから動作クロック値を選択し、選択した動作クロック値を上記プロセッサへ設定するクロック決定処理とを備えることを特徴とする省電力制御を計算機で実行させるプログラム。

【請求項10】 複数のタスクを動作させるプロセッサの動作クロックを制御する省電力制御を計算機で実行させるプログラムを記録した記録媒体において、複数の優先度と、上記複数の優先度それぞれに対応する優先度による実行を待つタスクの数とを優先度テーブルへ格納する優先度テーブル格納処理と、上記プロセッサへ設定することができる複数の動作クロック値をクロックテーブルへ格納するクロックテーブル格納処理と、

上記優先度テーブルから上記複数の優先度と上記タスクの数とを読み出し、読み出した複数の優先度とタスクの数とを用いて、上記プロセッサにかかる負荷値を算出し、算出した負荷値を用いて、上記クロックテーブルから動作クロック値を選択し、選択した動作クロック値を上記プロセッサへ設定するクロック決定処理とを備えることを特徴とする省電力制御を計算機で実行させるプログラムを記録した計算機で読み取り可能な記録媒体。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】 この発明は、ダイナミッククロック制御を持つプロセッサでの省電力制御方式に関する。特に、優先度をベースとしたタスクの実行制御を行

うオペレーティングシステムにおいて、タスクの動作状況に応じて適切なCPUクロックを選択する方式に関する。

【0002】

【従来の技術】ダイナミックに動作クロックを変更可能なプロセッサに対して、その動作クロックを決定する方法として、特開平9-297688や特開平10-143274に記載の方法がある。これらは、タスクが処理を行うのに必要な実行時間を基準にして動作クロックを決定している。また、特開平9-297688では、処理に必要な実行時間から求められた動作クロックが最も高いタスクに高い優先度を付けていく、ということも成されている。

【0003】

【発明が解決しようとする課題】しかし、これらの方法では、処理の開始から完了までに時間の制約があるリアルタイムシステムでは、動作クロックを処理に必要な実行時間から算出すると不都合が生じる場合がある。例えば、高い優先度を設定されたタスクでも、必要な実行時間が短ければ、低い動作クロックが算出され、低い動作クロックで実行されてしまうため、結果として時間制約以内に処理を完了できない、という問題があった。また、一般的なリアルタイムシステムを構成するタスクとして、高い優先度を持つタスク程、必要な処理時間が短い傾向にあり、この点を考慮して、動作クロックを選択する必要があるという問題があった。

【0004】この発明は、リアルタイムシステムにおけるタスク実行時間の制約を考慮しつつ、省電力化を実現する制御方式及び制御方法を提供することを目的とする。

【0005】

【課題を解決するための手段】この発明に係る省電力制御方式は、複数のタスクを動作させるプロセッサの動作クロックを制御する省電力制御方式において、複数の優先度と、上記複数の優先度それぞれによる実行を待つタスクの数とを保持する優先度テーブルと、上記プロセッサへ設定することができる複数の動作クロック値を保持するクロックテーブルと、上記優先度テーブルから上記複数の優先度と上記タスクの数とを読み出し、読み出した複数の優先度とタスクの数とを用いて、上記プロセッサにかかる負荷値を算出し、算出した負荷値を用いて、上記クロックテーブルから動作クロック値を選択し、選択した動作クロック値を上記プロセッサへ設定するクロック決定部とを備えることを特徴とする。

【0006】上記省電力制御方式は、さらに、上記プロセッサの動作クロックを決めるための基準値を保持する基準値テーブルを備え、上記クロック決定部は、上記基準値テーブルから基準値を読み出し、上記基準値と上記負荷値とを用いて、動作クロックを選択することを特徴とする。

【0007】上記優先度テーブルは、さらに、上記複数の優先度それぞれに対応する重み付け値を保持し、上記クロック決定部は、上記複数の優先度と上記タスクの数と上記重み付け値とを用いて負荷値を算出することを特徴とする。

【0008】上記優先度テーブルは、さらに、上記複数の優先度それぞれに対応する設定クロック値を保持し、上記クロック決定部は、上記負荷値を算出する場合に、上記設定クロック値を用いることを特徴とする。

10 【0009】上記省電力制御方式は、さらに、実行を要求するタスク名と実行を終了したタスク名とのいずれかを入力し、入力したタスク名から優先度を取得し、取得した優先度を用いて、上記優先度テーブルに保持するタスクの数を更新するタスク管理部を備えることを特徴とする。

【0010】上記省電力制御方式は、上記タスク管理部が優先度テーブルに保持するタスク数を更新した場合に、上記クロック決定部を起動することを特徴とする。

20 【0011】上記省電力制御方式は、所定の周期で上記クロック決定部を起動することを特徴とする。

【0012】この発明に係る省電力制御方法は、複数のタスクを動作させるプロセッサの動作クロックを制御する省電力制御方法において、複数の優先度と、上記複数の優先度それぞれに対応する優先度による実行を待つタスクの数とを優先度テーブルへ格納する優先度テーブル格納工程と、上記プロセッサへ設定することができる複数の動作クロック値をクロックテーブルへ格納するクロックテーブル格納工程と、上記優先度テーブルから上記複数の優先度と上記タスクの数とを読み出し、読み出した複数の優先度とタスクの数とを用いて、上記プロセッサにかかる負荷値を算出し、算出した負荷値を用いて、上記クロックテーブルから動作クロック値を選択し、選択した動作クロック値を上記プロセッサへ設定するクロック決定工程とを備えることを特徴とする。

30 【0013】この発明に係る省電力制御を計算機で実行させるプログラムは、複数のタスクを動作させるプロセッサの動作クロックを制御する省電力制御を計算機で実行させるプログラムにおいて、複数の優先度と、上記複数の優先度それぞれに対応する優先度による実行を待つタスクの数とを優先度テーブルへ格納する優先度テーブル格納処理と、上記プロセッサへ設定することができる複数の動作クロック値をクロックテーブルへ格納するクロックテーブル格納処理と、上記優先度テーブルから上記複数の優先度と上記タスクの数とを読み出し、読み出した複数の優先度とタスクの数とを用いて、上記プロセッサにかかる負荷値を算出し、算出した負荷値を用いて、上記クロックテーブルから動作クロック値を選択し、選択した動作クロック値を上記プロセッサへ設定するクロック決定処理とを備えることを特徴とする。

40 【0014】この発明に係る省電力制御を計算機で実行

させるプログラムを記録した計算機で読み取り可能な記録媒体は、複数のタスクを動作させるプロセッサの動作クロックを制御する省電力制御を計算機で実行させるプログラムを記録した記録媒体において、複数の優先度と、上記複数の優先度それぞれに対応する優先度による実行を待つタスクの数とを優先度テーブルへ格納する優先度テーブル格納処理と、上記プロセッサへ設定することができる複数の動作クロック値をクロックテーブルへ格納するクロックテーブル格納処理と、上記優先度テーブルから上記複数の優先度と上記タスクの数とを読み出し、読み出した複数の優先度とタスクの数とを用いて、上記プロセッサにかかる負荷値を算出し、算出した負荷値を用いて、上記クロックテーブルから動作クロック値を選択し、選択した動作クロック値を上記プロセッサへ設定するクロック決定処理とを備えることを特徴とする。

【0015】

【発明の実施の形態】一般的なリアルタイムシステムを構成するタスクとして、高い優先度を持つタスク程、必要な処理時間が短い傾向にあるという点を考慮して動作クロックを選択する必要がある。この発明では、各優先度毎に優先度の高さに応じたCPU動作クロックを左右する情報を保持することによって、優先度の高いタスクの存在数が多いほど高いCPU動作クロックを選択可能とし、リアルタイムシステムにおける時間制約を保持しつつ、最小の動作クロックを選択することでプロセッサの省電力を実現する省電力制御方式及び方法を提供する。以下、実施の形態の一例を説明する。

【0016】実施の形態1. 図1は、本発明の省電力制御方式の機能を表すブロック図の一例である。図1において、101は本発明が実際に搭載されるオペレーティングシステム全体である。オペレーティングシステム101の内部には、本発明を実施するにあたり、最低限必要な機能が搭載されており、以下の構成要素を含む。

【0017】102はオペレーティングシステムが管理する実行主体であるタスクの情報・状態を管理・制御するタスク管理部である。103はタスク管理部102の中にあって、CPUによる実行を待っているタスクをリストの形式で管理しているレディーキュー及びレディキューを操作するための機能である。104はCPU (Central Processing Unit) 外部にあるタイムハードウェアからの割込みをもとに実行され、オペレーティングシステム101内の時刻やタスクに係わる様々な時間関係の機能（例えば、タイムアウトや周期的な動作等）を実現する時間管理部である。

【0018】105はCPUが取り得る動作クロックの一覧を示すCPUクロックテーブル（クロックテーブル）である。106は前記CPUクロックテーブル105の各動作クロックを選択するための基準値を示すクロック境界値テーブルである。「クロック境界値テーブ

ル」は、「境界値テーブル」又は「基準値テーブル」ともいう。107はタスク管理部102内のレディキューにて保持される情報を元に現在のCPU負荷を数値化し、クロック境界値テーブル106より、CPU負荷に適したCPU動作クロックを選択し、実際にCPUの動作クロックを変更するために、CPUクロック制御用ハードウェア108にアクセスする、CPUクロック決定部である。

【0019】図2は、実施の形態1における優先度テーブル201、レディキュー202、境界値テーブル204、CPUクロックテーブル203の内容例を表す図である。図2において、一般的なリアルタイムOSと同じく、レディキューは各優先度毎にリストを構成しており、全リストを配列として管理している優先度テーブル201には、そのリストに繋がれているタスクの優先度を示すprior、リストの先頭のタスクを指示するPointer、レディキューのリストにつながれているタスクの個数を示すlist_noが格納されている。なお、一般的なリアルタイムOSでは、優先度テーブル201内に通常はpriorの領域はなく、配列の添え字にて管理しているが、ここでは説明の便宜上、優先度テーブル201内に情報を持っているものとしている。なお、201において優先度は数値が小さいほど高優先の処理である。つまり、prior=1が最も高優先で、prior=4が最も低優先のタスクである。

【0020】202は優先度テーブル201のPointerにて先頭を示されるレディキューリスト（レディキューともいう）であり、実行待ち状態の全てのタスクがいずれかのレディキュー202につながれている。また、レディキュー202は、一般にFIFO (First In First Out) で管理されている。つまり、後から追加されたタスクはリストの最後尾に繋がれ、リストの先頭のタスクからCPUが割り当てられていくことになる。

【0021】203はCPUが取り得る動作クロックの一覧が保持されたCPUクロックテーブルで、本実施の形態では30, 40, 50, 60のクロックがダイナミックに設定できることを示している。通常のCPUでは、これらの数値の単位はメガヘルツである。204はクロック境界値テーブル204であり、203で示した4段階の動作クロック間での閾値を、CPUの負荷値として示している。例えば、クロック境界値テーブル204での数値5は、CPUのクロックを30と40のいずれかにするかを決めるための数値で、CPU負荷値が0～4ならばクロックは30、5以上ならばクロックは40に設定することを示している。また、クロック境界値テーブル204の他の数値10, 15も同様な意味を示す。

【0022】図3は、実施の形態1におけるCPUクロック決定部の処理を示すフロー図である。図3におい

て、`prio_max`とは、優先度テーブル201にて管理する優先度の最大数値を示し、図2のテーブル構成においては、`prio_max = 4`である。また、`prioi`、`list_noi`は、優先度テーブルにおいて、優先度テーブル201の配列としてのインデックスⁱの要素中の`prio`及び`list_no`の値を示し、図2のテーブル構成においては、 $i = 0$ ならば201aを示し、 $prio^i = 1$ 、`list_noi = 1`、 $i = 2$ ならば201cを示し、 $prio^i = 3$ 、`list_noi = 2`となる。

【0023】次に、図2及び図3に従って、実施の形態1でのCPUクロック決定部107の処理を説明する。CPUクロック決定部107は、インデックスⁱ、CPU負荷値の総和`Totai`とを変数として使用する。ステップ301は、CPUクロック決定部107内での内部データの初期化である。インデックスⁱ、CPU負荷値の総和`Totai`の初期化及びその他必要な初期化を行う。次に、ステップ302にてインデックスⁱの範囲チェックを行い、範囲内であれば優先度テーブル201においてインデックスⁱが有効であることが判る。その結果、範囲内であればステップ303において、インデックスⁱの優先度テーブル201の要素におけるCPU負荷値を、レディキュー202に繋がれているタスクの数と優先度値を乗じた値として計算し、全てのレディキュー202でのCPU負荷値の総和を示す`Totai`に加える。その後、ステップ304にてインデックスを1つ進めて、ステップ302へ戻る。ステップ302～304を繰り返すことにより、全てのレディキュー202に対するCPU負荷値の総和が`Totai`に入り、ステップ305へ進む。

【0024】なお、図2のテーブル構成では、ステップ303にて`Totai`に加えられるCPU負荷値は、201aにて4、201bにて0、201cにて4、201dにて1となり、ステップ305へ進んだ時点では、`Totai = 9`となっている。次に、ステップ305では、上記`Totai`値とクロック境界値テーブル204を比較し、ステップ306にてCPUを動作させるべきクロック値をCPUクロックテーブル203から取り出し、`clock`へ入れる。ステップ307では、求められた`clock`をCPUクロック制御用ハードウェア108に設定してCPUの動作クロックを変更し、CPUクロック決定部107の処理を終了する。

【0025】なお、ステップ305及び306でのクロック境界値テーブル204との比較及びCPUクロックテーブル203からのクロック値取り出しは、前記`Totai`を求めるためのループ処理302～304の中に追加してもよく、同じ結果が得られる。

【0026】また、境界値テーブルに保持する基準値は、計算機の稼動状況によって変更することも可能である。利用者が基準値を変更するためのインターフェースが

用意されていることが望ましい。

【0027】以上の処理により、レディキュー202に優先度の高いタスクが多い程、高いCPUクロックにて動作させることができ、一般に優先度の高いタスクには厳しい処理時間制約が付くリアルタイムシステムにおいて、適切なCPU動作クロックを求めることができる。また、比較的優先度の低いタスクが動作する場合には、電力消費の少ない低速なクロックで動作させることができるため、処理の重要度・緊急度と省電力の適切な調整を取ることができる。

【0028】以上のように、この実施の形態の省電力制御方式は、以下の構成要素、データを記憶するテーブルとを備え、タスクの優先度を考慮して、最適なプロセッサ動作クロックを選択することを特徴とする。

(a) レディキュー内で同一優先度を持つタスクの数を付与した、優先度テーブル。

(b) プロセッサが取り得る動作クロックの値を保持する、CPUクロックテーブル。

(c) プロセッサの各動作クロックを決めるための基準値を保持する、境界値テーブル。

(d) 上記優先度テーブルの優先度値、同一優先度タスク数より、プロセッサの現在の負荷値を計算し、境界値テーブルとの比較により、現在のプロセッサ負荷に最適なCPU動作クロックを設定する、CPUクロック決定部。

【0029】実施の形態2。次に、実施の形態2について説明する。この実施の形態の省電力制御方式の構成は、図1で表したブロック図と同様である。図4は、実施の形態2における優先度テーブル401、レディキュー202、境界値テーブル204、CPUクロックテーブル203の内容例を表す図である。図4において、202～204は、図2と同じものである。401は図2の201と同様な優先度テーブルであるが、その構成要素として、各優先度毎にCPU負荷値を計算する際の重み付け値Wが格納されている。

【0030】図5は、実施の形態2におけるCPUクロック決定部の処理を示すフロー図である。図5において、`prio_max`、`prioi`、`list_noi`は、図3にて記載のものと同じ意味を持っている。

【0031】次に、図4及び図5に従って、実施の形態2でのCPUクロック決定部107の処理を説明するが、図5におけるステップ301～302及びステップ304～307は、実施の形態1におけるものと同じであるため説明は省略する。ステップ503では、レディキュー202毎のCPU負荷値を求め、`Totai`に加えているが、レディキュー202毎のCPU負荷値の計算に各優先度毎に設定された重み付けWが乗せられている。ここで W^i とは、インデックスⁱで示された優先度テーブル401内のWの値であり、 $i = 0$ ならば401aのWである10を、 $i = 3$ ならば401dのWである

1を示す。ステップ302, 503, 304のループ処理の結果、図4でのテーブル構成においては、CPUの総負荷値Totalは15となり、クロック境界値テーブル204及びCPUクロックテーブル203により、ステップ306におけるclockには、60が設定されることになる。

【0032】なお、ステップ305及び306でのクロック境界値テーブル204との比較及びCPUクロックテーブルからのクロック値取り出しへ、前記Totalを求めるためのループ処理302, 503, 304の中に追加してもよく、同じ結果が得られる。

【0033】また、優先度テーブルに保持する重み付け値Wは、計算機の稼動状況によって変更することも可能である。利用者が重み付け値Wを変更するためのインターフェースが用意されていることが望ましい。

【0034】以上の処理により、レディキューに優先度の高いタスクが多い程、高いCPUクロックにて動作させることができるとともに、優先度毎にCPU負荷値への重み付けを設定するために、リアルタイムシステムの実状やタスクへの優先度の割り付け具合に合わせた、処理の重要度・緊急度と省電力の適切な調整を取ることができる。

【0035】以上のように、この実施の形態では、実施の形態1の省電力制御方式に加え、優先度テーブルに、各優先度毎に負荷の重み付けを付与し、この優先度毎の重み付けをプロセッサの負荷値計算に入れたことを特徴とする。

【0036】実施の形態3。次に、実施の形態3について説明する。この実施の形態の省電力制御方式の構成は、図1で表したブロック図と同様である。図6は、実施の形態3における優先度テーブル601、レディキュー202、CPUクロックテーブル203の内容例を表す図である。なお、この実施の形態では、境界値テーブルは必要としない。図6において、202～203は、図2と同じものである。601は図2の201と同様な優先度テーブルであるが、その構成要素として、各優先度毎に動作すべきCPUクロック値Limitが格納されている。

【0037】図7は、実施の形態3におけるCPUクロック決定部の処理を示すフロー図である。図7において、`prio_max`, `list_no`は、図3にて記載のものと同じ意味を持っている。`Limit`は、インデックスiで示された優先度テーブル601内の`Limit`の値(設定クロック値)であり、`i=0`ならば601aの`Limit`の値である60を、`i=3`ならば601dの`Limit`である30を示す。

【0038】次に、図6及び図7に従って、実施の形態3でのCPUクロック決定部107の処理を説明する。ステップ701は、以下の処理のための初期化である。ステップ702では、インデックスiの範囲チェックを

行い、ステップ703では、i番目のレディキュー202にタスクが存在しているかをチェックしている。ステップ703にてレディキュー202にタスクが存在しない場合は、ステップ704に進み、次のレディキュー202をチェックするためにインデックスiを1つ進め、ステップ702へ戻る。ステップ703にてレディキュー202にタスクが存在する場合は、ステップ705へ進む。ステップ705では、インデックスiでしめされた優先度テーブル601の`Limit`値をclockへ入れ、ステップ706にてCPUクロック制御用ハードウェア108へ値を設定する。

【0039】なお、ステップ703にてタスクが存在しているレディキュー202が見つかった場合にループ処理を終了することは、通常のシステムでは優先度が高い処理ほど高速なCPUクロックを必要とするため、インデックスi=0から順にチェックして、最初にステップ703にてNo判断となることは、タスクの存在する最も優先度の高いレディキュー202であることを示し、最も高速なCPUクロックが要求されている個所を検出することになるということに依る。

【0040】以上の処理により、レディキュー202に優先度の高いタスクが多い程、高いCPUクロックにて動作させることができるとともに、優先度毎に動作させるべきCPUクロック値を設定するために、リアルタイムシステムの実状やタスクへの優先度の割り付け具合に合わせた、処理の重要度・緊急度と省電力の適切な調整を取ることができる。

【0041】また、優先度テーブルに保持する設定クロック値(Limitの値)は、計算機の稼動状況によって変更することも可能である。利用者が設定クロック値を変更するためのインターフェースが用意されていることが望ましい。

【0042】以上のように、この実施の形態の省電力制御方式は、優先度テーブルに、各優先度毎に予め設定された、動作すべきCPUクロック値を付与し、この優先度毎に設定されたCPUクロックで動作させることを特徴とする。

【0043】実施の形態4。次に、実施の形態4について説明する。この実施の形態の省電力制御方式の構成は、図1で表したブロック図と同様である。図8は、実施の形態4における優先度テーブル801、レディキュー202、境界値テーブル204、CPUクロックテーブル203の内容例を表す図である。図8において、202～204は、図2と同じものである。但し、レディキュー202においてタスクのつながり方は図2とは異なる。801は図2の201と同様な優先度テーブルであるが、その構成要素として、各優先度毎にCPU負荷値を計算する際の重み付け値Wと、各優先度毎に動作すべきCPUクロック値Limitが格納されている。

【0044】図9は、実施の形態4におけるCPUクロ

ック決定部の処理を示すフロー図である。図9において、`prio_max`, `prioi`, `list_noi`及び`Limiti`は、図3及び図7にて記載のものと同じ意味を持っている。

【0045】次に、図8及び図9に従って、実施の形態4でのCPUクロック決定部の処理を説明する。なお、図9におけるステップ304, 306～307は実施の形態1におけるものと同じであり、ステップ503は実施の形態2におけるものと同じであるため説明を省略する。ステップ901は以下の処理のための初期化である。ステップ902ではインデックス*i*の範囲チェックを行い、ステップ903では、*i*番目のレディキュー202にタスクが存在しているかをチェックしている。ステップ904は、タスクの存在するレディキュー202のうち、`Limit`で要求している最大のCPUクロック値を求めるための処理である。ステップ902から304のループ処理により、タスクが存在するレディキュー202の中で、要求クロックの最大値`Base`及びCPU負荷値`Total`が求められステップ907へ進む。

【0046】ステップ907では、`Base`のCPUクロックを必要とする最小のCPU負荷値、つまりはクロック境界値テーブル204内の適切な値を求める。例えば、`Base = 50`であるならば10が、`Base = 40`ならば5が得られる。次に、ステップ908では、ステップ907にて求めたCPU負荷値と902～304のループ処理で求められた`Total`を足し、各優先度毎に設定された`Limit`を考慮したCPU負荷値が求められる。次に、ステップ306, 307にて、ステップ908で求めた`Total`をもとに動作すべきCPUクロックを求め、CPUクロック制御用ハードウェア108に設定することになる。

【0047】なお、この実施の形態4では、優先度テーブル801は重み付け値Wを有するが、実施の形態3で示した優先度テーブル601のように、重み付け値Wを有していない優先度テーブルを用いて、CPU負荷値を算出することも可能である。

【0048】以上のように、この実施の形態では、実施の形態1の省電力制御方式に加え、優先度テーブルに、各優先度毎に、負荷の重み付け、及び予め設定された動作すべきCPUクロック値を付与したことを特徴とする。

【0049】実施の形態5。なお、本発明における別の実施の形態の処理フローを、図10に示す。図10と図9の相違点は、ステップ1001にて`Limiti`の0チェックを行い、もし0でなければ（つまり、`Limiti`を設定していたら）ループ処理を終了してステップ907へ進む、という点である。これは、通常のリアルタイムシステムでは、全ての処理に時間的制約が課されているわけではないため、`Limiti`を設定する必要のな

いタスクも多数存在することに起因している。

【0050】以上の実施の形態4, 5での処理により、レディキューに優先度の高いタスクが多い程、高いCPUクロックにて動作させることができるとともに、優先度毎にCPU負荷値への重み付けの設定や、優先度毎に動作させるべきCPUクロック値を設定するために、リアルタイムシステムの実状やタスクへの優先度の割り付け具合に合わせた、処理の重要度・緊急度と省電力の適切な調整を取ることができる。

10 【0051】実施の形態6。次に、実施の形態5について説明する。この実施の形態の省電力制御方式の構成は、図1で表したブロック図と同様である。図11は、実施の形態5におけるCPUクロック決定部を呼び出すレディキューへの挿入及び削除機能の処理フローを表す図である。

【0052】キューへの挿入処理では、ステップ1101にて、挿入するタスクの優先度から優先度テーブルの位置を求める。一般には、優先度テーブルは配列をなし、優先度も数値が小さいほど高優先であるため、

20 【(タスクの優先度) - 1】が、優先度テーブルの配列としてのインデックスとなっている。なお、優先度テーブルのインデックス0をリザーブとして、タスクの優先度がそのまま優先度テーブルのインデックスにしている場合もある。この実施の形態では、テーブルが0(零)オリジンの場合を想定している。次に、ステップ1102では、1101で求められた優先度テーブルによってポイントされているレディキューの末尾にタスクを挿入し、ステップ1103にてCPUクロック決定部107が呼び出される。なお、ステップ1102では、レディキューにタスクを挿入するとともに、実施の形態1～5における`list_noi`の値をインクリメントすることは言うまでも無い。

30 【0053】キューからの削除処理では、ステップ1104にて削除するタスクの優先度から優先度テーブルの位置を求める。優先度テーブルの位置の求め方は、キューへの挿入処理と同じである。次に、ステップ1105にてレディキューからタスクを削除し、ステップ1106にてCPUクロック決定部を呼び出す。なお、ステップ1105では、レディキューからタスクを削除するとともに、実施の形態1～5における`list_noi`の値をデクリメントすることは言うまでも無い。

40 【0054】次に、実施の形態6におけるCPUクロック決定部の処理フローを図12に沿って説明する。図12は、図2に示す実施の形態1で用いた優先度テーブル201、レディキュー202、クロック境界値テーブル204、CPUクロックテーブル203を用いて動作するCPUクロック決定部107の処理フロー図を示している。なお、実施の形態2～5の場合はやや異なる処理フローとなるが、図5、図7、図9、図10より類推可能である。

【0055】図12において、ステップ305～307は、図3の同じステップ番号のものと同じである。ステップ1201にて本処理フローが、レディキュー202への挿入時と、レディキュー202からの削除時とのいずれから呼び出されたかを判定する。判定の基準は、CPUクロック決定部に渡されるパラメータにて可能である。キュー挿入処理から呼び出された場合はステップ1202へ進み、挿入されたタスクの優先度によるCPU負荷値分をTotailへ加算する。一方、キュー削除処理より呼び出された場合は、ステップ1203へ進み、削除されたタスクの優先度によるCPU負荷値分をTotailより減算する。以降、ステップ305～307によって、ステップ1202, 1203によって再計算されたTotail値にてCPUの動作クロックを決定し、ハードウェアへ設定する。

【0056】以上の処理により、CPUの動作クロックを決定するCPU負荷値を、CPU負荷値を左右するレディキューの操作時に行なうことで、CPU負荷値の変化を即座にCPUクロックへ反映可能とともに、CPUクロック決定部107の処理を軽減することができる。

【0057】以上のように、この実施の形態では、実施の形態1から実施の形態4の省電力制御方式に加え、前記CPUクロック決定部を、レディキュー操作時に実行させることにより、CPUクロック決定部の処理を軽減することを特徴とする。

【0058】実施の形態7. 次に、実施の形態7について説明する。この実施の形態の省電力制御方式の構成は、図1で表したブロック図と同様である。図13は、実施の形態7におけるオペレーティングシステム内の時間管理部104のタイム109からの割込みが発生した時の割込み処理フローを示している。ステップ1301では、通常のオペレーティングシステムにおけるタイム割込み発生時の処理を行う。例えば、タスクのタイムアウト検出等である。次に、ステップ1302にてCPUクロック決定部107を呼び出すことにより、通常のオペレーティングシステムが保有している、一定時間間隔で発生するタイム割込み処理を利用して、CPUの動作クロックを再計算する。CPUクロック決定部の処理フローは、図3、図5、図7、図9、図10のいずれかと同じである。

【0059】以上の処理により、CPUの動作クロックを決定するCPU負荷値を、一定時間間隔で再計算することにより、非常に発生頻度の高いレディキュー操作時の再計算を不要として、オペレーティングシステムの負荷を軽減することができる。

【0060】以上のように、この実施の形態では、実施の形態1から実施の形態4の省電力制御方式に加え、前記CPUクロック決定部を、一定時間間隔で周期的に実行させることにより、レディキュー操作時の処理追加を

不要とすることを特徴とする。

【0061】実施の形態8. 次に、実施の形態8について説明する。実施の形態8は、図11におけるキュー挿入時のフロー及び図13のタイム割込みを利用した一定時間間隔でのCPU負荷値再計算のフローで構成される。

【0062】以上の処理により、実施の形態7におけるオペレーティングシステムの負荷軽減の効果を保持しつつ、レディキューへのタスク挿入時にもCPU負荷値を再計算することにより、CPU負荷の増加へ即時に対応可能となり、リアルタイム処理への応答性低下も回避することができる。

【0063】以上のように、この実施の形態では、実施の形態1から実施の形態4の省電力制御方式に加え、前記CPUクロック決定部を、レディキュー操作時及び一定時間間隔で周期的に実行させることにより、レディキュー操作時の処理追加を軽減させるとともに、CPUクロック決定部の処理量を軽減させることを特徴とする。

【0064】
20 【発明の効果】この発明によれば、タスクの優先度と、待ち状態にあるタスクの数とを考慮してタスクの動作クロックを決定することができる。このため、複数のタスクが待ち状態にある場合にも柔軟に対応して動作クロックを決定することができる。

【0065】また、境界値テーブルを備えることにより、少なくともタスクの優先度と待ち状態にあるタスクの数とを要素として算出した負荷値を用いて動作クロックを決定することができる。また、基準値を計算機の稼動状況に応じて変更することにより、タスクの実行状況を改善することを容易にできる。

【0066】また、優先度テーブルに重み付け値を有することにより、タスクの動作クロックを利用者の意図に応じて決定することができる。また、重み付け値を変更することにより、計算機の稼動状況に合わせた動作クロックを決定することができる。

【0067】また、予め、優先度テーブルに設定クロック値(Limitの値)を有することにより、計算機の稼動状況やタスクの優先度の割付具合に応じた調整をすることができる。

【0068】また、タスク管理部を有することにより、タスクの開始、終了を優先度テーブルに反映することができるとなり、より適切な動作クロックの決定に寄与することになる。

【0069】また、優先度テーブルのタスクの数が更新された場合にクロック決定部を起動することにより、タスクの実行状況に対応した動作クロックを変更することができる。

【0070】また、クロック決定部を定期的に起動することにより、計算機の稼動状況に応じて定期的にプロセッサの動作クロックを更新することができる。

【図面の簡単な説明】

【図1】 本発明の省電力制御方式の機能を表すブロック図。

【図2】 実施の形態1における優先度テーブル201、レディキュー202、境界値テーブル204、CPUクロックテーブル203の内容例を表す図。

【図3】 実施の形態1におけるCPUクロック決定部の処理を示すフロー図。

【図4】 実施の形態2における優先度テーブル401、レディキュー202、境界値テーブル204、CPUクロックテーブル203の内容例を表す図。

【図5】 実施の形態2におけるCPUクロック決定部の処理を示すフロー図。

【図6】 実施の形態3における優先度テーブル601、レディキュー202、CPUクロックテーブル203の内容例を表す図。

【図7】 実施の形態3におけるCPUクロック決定部の処理を示すフロー図。

【図8】 実施の形態4における優先度テーブル801、レディキュー202、境界値テーブル204、CPUクロックテーブル203の内容例を表す図。

* 【図9】 実施の形態4におけるCPUクロック決定部の処理を示すフロー図。

【図10】 実施の形態5におけるCPUクロック決定部の処理フローを示す図。

【図11】 実施の形態5におけるCPUクロック決定部を呼び出すレディキューへの挿入及び削除機能103の処理フローを表す図。

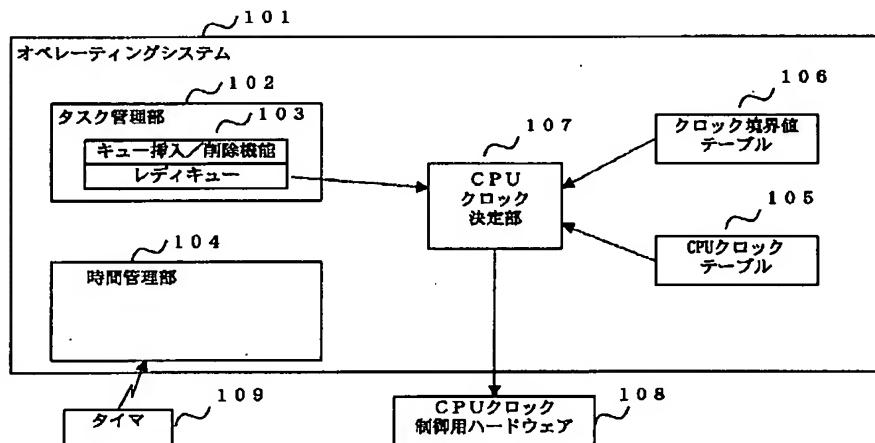
【図12】 実施の形態6におけるCPUクロック決定部の処理フロー図。

10 【図13】 実施の形態7におけるオペレーティングシステム内の時間管理部104のタイマハードウェア109からの割込みが発生した時の割込み処理フロー図。

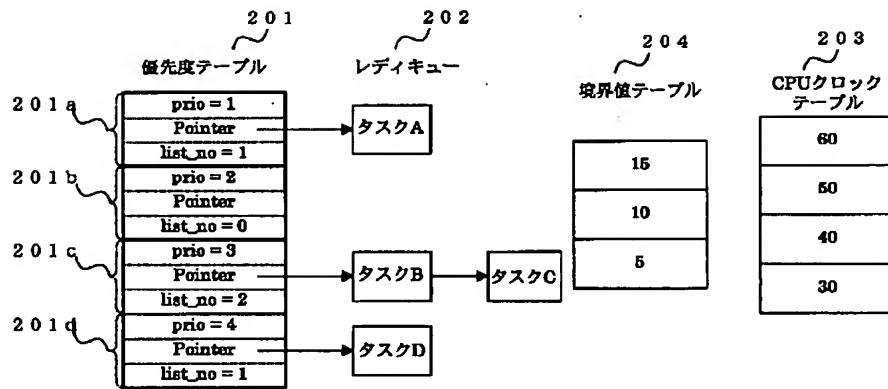
【符号の説明】

101 オペレーティングシステム、102 タスク管理部、103 レディキュー及びレディキューを操作する機能、105 CPUクロックテーブル、106 クロック境界値テーブル、107 CPUクロック決定部、108 CPUクロック制御用ハードウェア、109 タイマ、201, 401, 601, 801 優先度テーブル、202 レディキュー、203 CPUクロックテーブル、204 境界値テーブル。

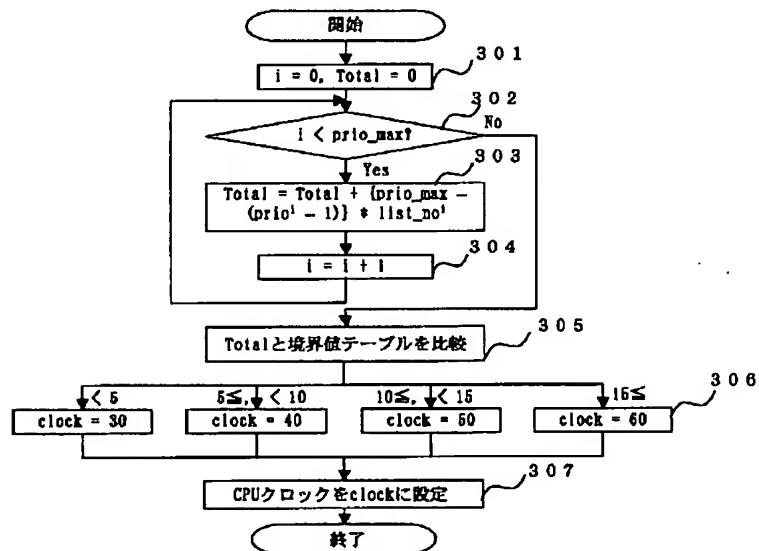
【図1】



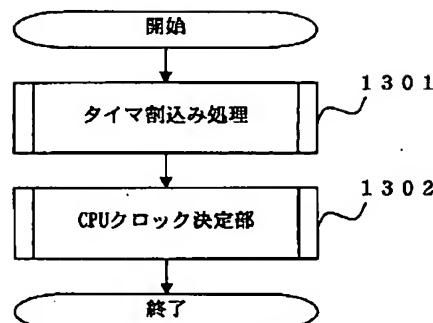
【図2】



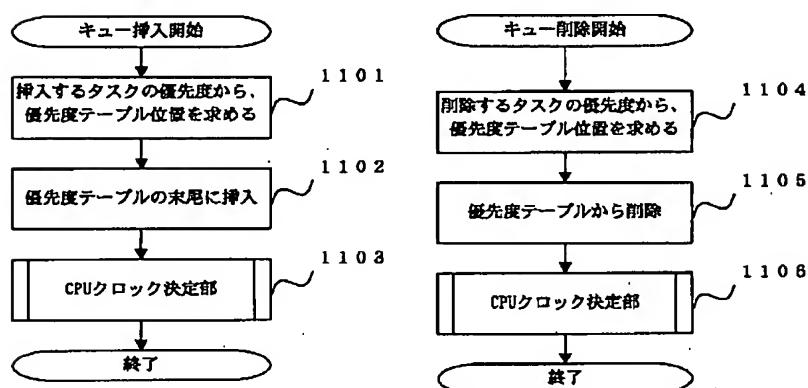
【図3】



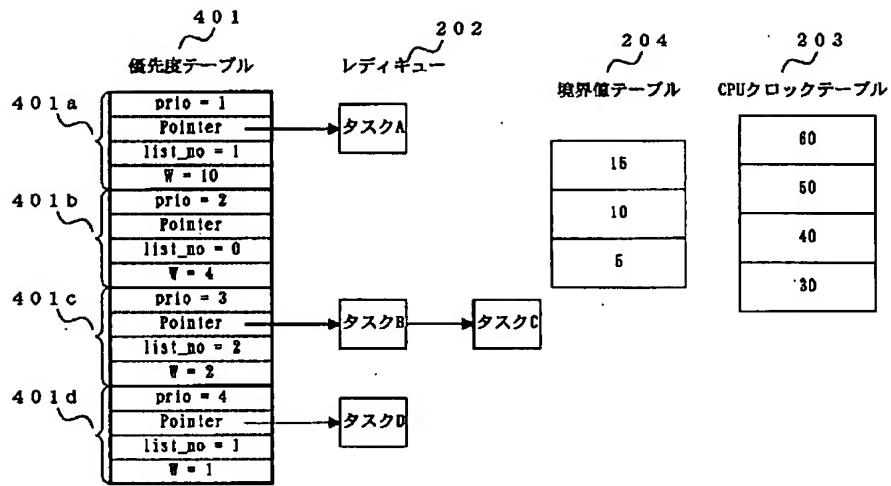
【図13】



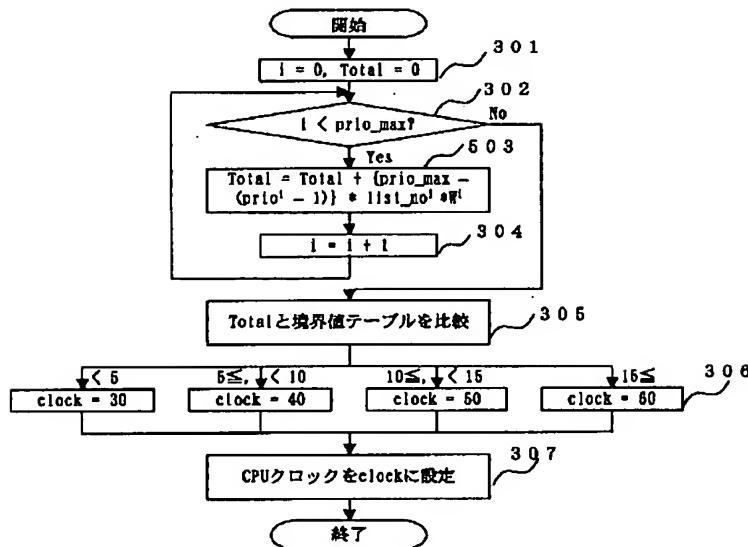
【図11】



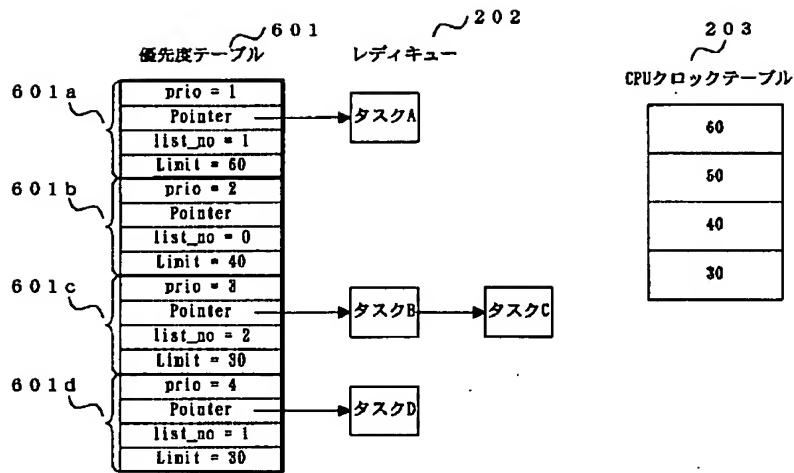
【図4】



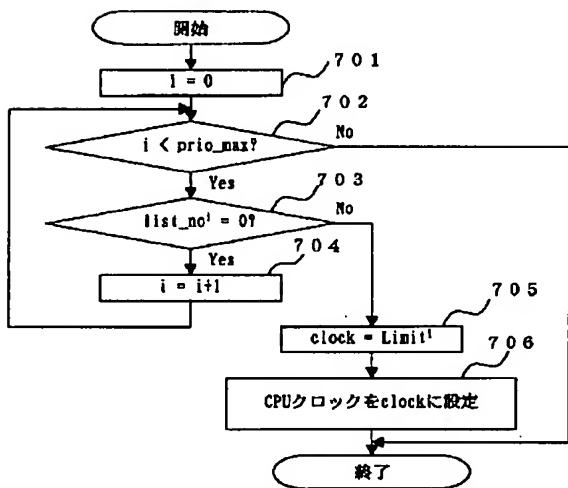
【図5】



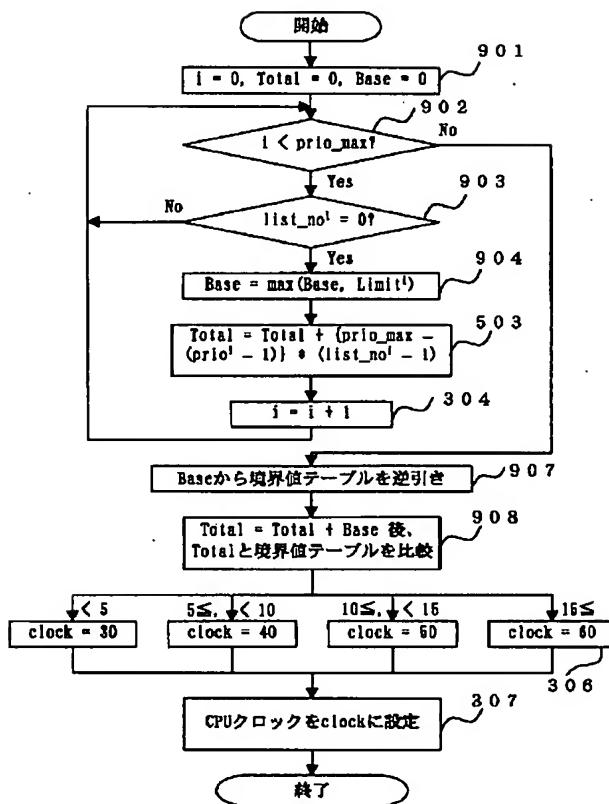
【図6】



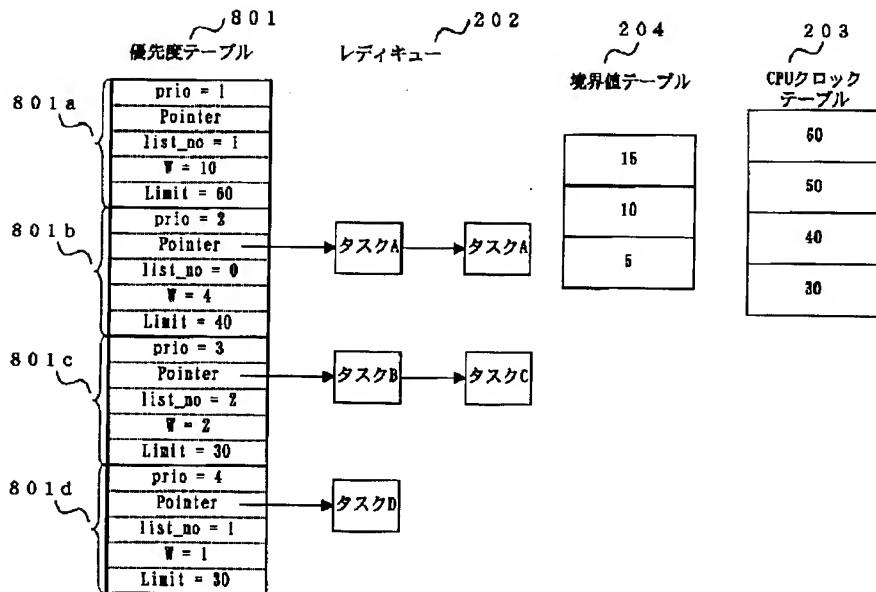
【図7】



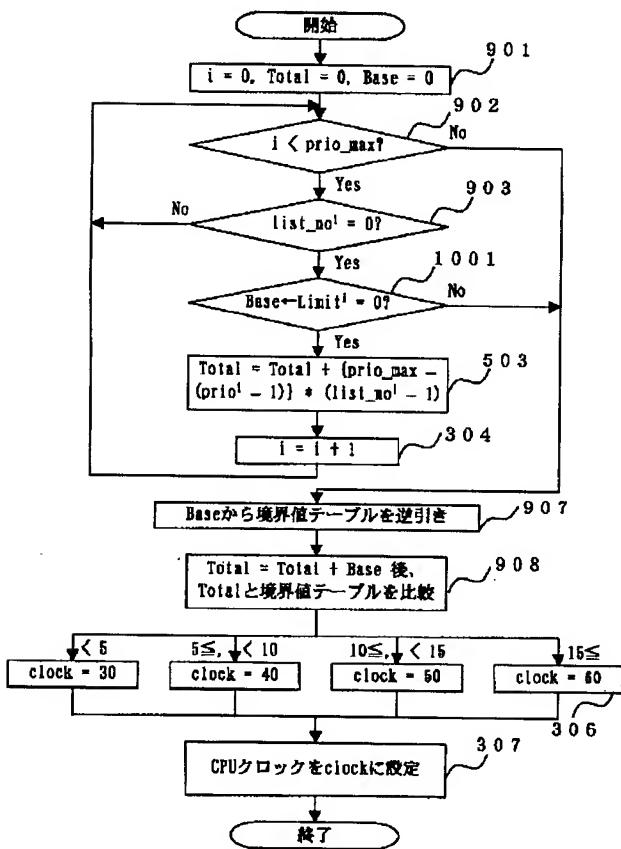
【図9】



【図8】



【図10】



【図12】

